

Docket No. 50277-2343

*Patent*

UNITED STATES PATENT APPLICATION

FOR

AUTOMATIC AND DYNAMIC PROVISIONING OF DATABASES

INVENTORS:

BENNY SOUDER  
JAMES STAMOS  
LIK WONG  
JOHN CIMINSKI  
ANAND LAKSHMINATH  
ALAN DOWNING

PREPARED BY:

HICKMAN PALERMO TRUONG & BECKER LLP  
1600 WILLOW STREET  
SAN JOSE, CALIFORNIA 95125  
(408) 414-1080

ASSIGNEE:

ORACLE INTERNATIONAL CORPORATION  
500 ORACLE PARKWAY  
REDWOOD SHORES, CA 94065

"Express Mail" mailing label number EV322192654US

Date of Deposit November 21, 2003

OID 2003-178-01

## AUTOMATIC AND DYNAMIC PROVISIONING OF DATABASES

### RELATED APPLICATIONS

[0001] This application claims priority to U.S. Provisional Application No. 60/495,368, entitled *Computer Resource Provisioning*, filed by Debashish Chatterjee, et al. on August 14, 2003, the contents of which are incorporated herein by reference; this application claims priority to U.S. Provisional Application No. 60/500,050, entitled *Automatic And Dynamic Provisioning Of Databases*, filed by Benny Souder, et al. on September 3, 2003, the contents of which are incorporated herein by reference; this application claims priority to U.S. Provisional Application 60/500,096, entitled *Service Based Workload Management and Measurement in a Distributed System*, filed by Carol Colrain, et al. on September 3, 2003, the contents of which are incorporated herein by reference.

[0002] The following applications are related to the current application:

U.S. Provisional Application No. 60/410,883, entitled *Oracle Streams*, filed by Alan Downing, et al. on September 9, 2002, the contents of which are incorporated herein by reference;

U.S. Provisional Application No. 60/400,532, entitled *Utilizing Rules in Distributed Information Sharing*, filed by Edwina Lu, et al. on August 1, 2002, the contents of which are incorporated herein by reference;

U.S. Application No. 10/449,873, entitled *Utilizing Rules in a Distributed Information Sharing System*, filed by Edwina Lu, et al. on May 30, 2003, the contents of which are incorporated herein by reference; and

U.S. Application No. 10/353,381, entitled *Pluggable Tablespaces For Database Systems*, filed by Juan R. Loaiza, et al. on January 28, 2003, the contents of which are incorporated herein by reference.

## FIELD OF THE INVENTION

[0003] The present invention relates to distributed database systems, and in particular, to allocating resources in a database system to meet varying workload demands.

## BACKGROUND OF THE INVENTION

[0004] Enterprises have data processing systems that serve the needs of many applications. A typical enterprise data processing system allocates individual resources for each of the enterprise's applications. Application work load characteristics are not; some applications are busy during the day; some others during the night; some reports are run once a week and some others once a month. Enough resources are acquired to ensure that the peak work load of each application may be handled. As a result, when the workload of an application is not at its peak, a lot of resource allocated to an application is left unutilized.

[0005] Base on the foregoing, there is need to provide a mechanism that meet vary workload requirements of applications in a way that reduces the need to allocate an overcapacity of resources that is left unused during non-peak load times.

[0006] The approaches described in this section are approaches that could be pursued, but not necessarily approaches that have been previously conceived or pursued. Therefore, unless otherwise indicated, it should not be assumed that any of the approaches described in this section qualify as prior art merely by virtue of their inclusion in this section.

## BRIEF DESCRIPTION OF THE DRAWINGS

[0007] The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

[0008] FIG. 1 is a block diagram depicting a distributed database system upon which an embodiment of the present invention may be implemented.

[0009] FIG. 2 is a block diagram of procedure used to automatically and dynamically provision tablespaces according to an embodiment of the present invention.

[0010] FIG. 3 is a flow chart depicting a process for automatically and dynamically provisioning tablespaces according to an embodiment of the present invention.

[0011] FIG. 4 is a block diagram depicting a mechanism for synchronizing tablespaces that may be automatically and dynamically provisioned according to an embodiment of the present invention.

[0012] FIG. 5 is a flow chart for automatically and dynamically provisioning a tablespace and tablespace synchronization mechanism according to an embodiment of the present invention.

[0013] FIG. 6 is a block diagram depicting a computer system that may be used to implement an embodiment of the present invention.

## DETAILED DESCRIPTION OF THE INVENTION

**[0014]** A method and apparatus for automatic and dynamic provisioning of databases is described. In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the present invention.

**[0015]** Approaches described herein may be used for provisioning of databases that requires a bulk transfer of data within in a distributed computing environment, such as a grid. The approaches do not require the manual intervention of a DBA to, for example, transfer a tablespace between the file systems of operating systems. Instead, the tablespaces may be provisioned automatically and dynamically by a grid computing system whenever it determines the need to dynamically provision a database. In addition, as copies of tablespaces are provisioned, synchronization mechanisms can also be automatically provisioned to keep the tablespaces and their copies in sync.

## EXEMPLARY DISTRIBUTED DATABASE SYSTEM

**[0016]** FIG. 1 shows a distributed database system 101 that may be used to implement an embodiment of the present invention. Distributed database system 101 includes database server 112 and database server 122. Database server 112 manages access to database 114 and database server 122 manages access to database 124. A database server, such as database servers 112 and 122, is a combination of a set of integrated software components and an allocation of computational resources, such as memory and processes for executing the set of integrated software components on a processor, where the combination of software and computational resources are used for managing a database. Among other functions of

database management, a database server governs and facilitates access to a database, processing requests by database clients to access the database. The database clients of a database server may include other database servers. Database 110 is a collection of database objects. Database objects include any form of structured data. Structured data is data structured according to a metadata description defining the structure. Structured data includes relational tables, object tables, object-relational tables, and bodies of data structured according to the Extensible Markup Language ("XML"), such as XML documents.

**[0017]** Tables 119 and 129 are "online" because each of them is defined as being part of a database. Offline tablespaces, such as offline tablespaces 180, are tablespaces that are detached, that is, not defined as being part of a database. The approaches described herein can be used to provision copies of online tablespaces and offline tablespaces.

**[0018]** Database 114 is referred to as a local database with respect to database server 112 because database server 112 does not require another database server to access database 114 on database server 112's behalf. However, to access database 124, database server 112 must request access from database server 122, which accesses the data on behalf of database server 112. Therefore, database 124 is not a local database of database server 112.

**[0019]** As mentioned before, a tablespace is a collection of storage containers. According to an embodiment of the present invention, the storage containers are files defined and managed by an operating system. An operating system is software that controls and manages a computer element's resources. Other software that is executed on the computer element, herein "applications", are executed by processes that operate under the control of the operating system. The software components of a database server are an example of an application. Access by processes executing the applications to the resources is also controlled

by the operating system. Applications access the resources by calling operating system routines and utilities.

**[0020]** Files defined and managed by an operating system are stored in persistent storage, such as disk storage, and organized as a hierarchy of directories that contain files and other directories. A file system directly controlled by an operating system on a computer element is referred to as a local file system with respect to the operating system. The files within the file system are referred to as local files with respect to the operating system. A file system is directly controlled by a particular operating system if access to the file system does not require another operating system to access the file system on the particular operating system's behalf. Thus, a file system in a shared disk system may be controlled by more than one operating system.

**[0021]** An operating system on a computing element that controls an application or a process is referred to herein as a local operating system with respect to the application or process. Likewise, the application or process is referred to as a local application or local process with respect to the operating system.

**[0022]** Processes executing under the control of an operating system are associated with an operating system account established by operating system functions responsible for registering accounts. Initiating a process for a user requires that the user logon under an account using an operating system function for logging on users.

**[0023]** While an operating system controls the resources of a computing element, the control can be shared with applications, which share control jointly with and subordinate to an operating system. For example, database server 112 is comprised of multiple "database server processes" associated with an operating system account. Processes associated with this operating system account, which include the multiple database server processes, are granted

exclusive read and write privileges by the operating system of computer element 110 to directories in the local file system of the operating system. Access by the database server processes to data in the directories is managed by database server 112. Processes associated with other operating system accounts may not access these directories. In this way, database server 112 shares joint and subordinate control over these directories and the files within them.

**[0024]** An operating system provides various functions and utilities that allow resources to be managed and handled in a certain way. For example, the operating system of computer element 110 can treat disk drives as a Redundant Array of Independent (or Inexpensive) Disks. Likewise, a database server can provide similar capabilities.

**[0025]** Database transport procedures 150 include routines that may be invoked by a database server to perform a specific set of steps needed to provision a tablespace. A procedure may perform some or all the steps needed to provision a tablespace, including steps conventionally and manually performed by a DBA. The procedures may be invoked to automatically and dynamically provision a database.

#### Data Transport Mechanisms

**[0026]** Database servers 112 and 122 transmit data between each other using various data transport mechanisms ("transport mechanisms"). These transport mechanisms include Database Server to Database Server file transport mechanism 130 ("DB file transport mechanism") and rule-based messaging system 140. For purposes of exposition, DB file transport mechanism 130 and rule-based messaging system 140 are depicted in Fig. 1 using blocks that are separate and distinct from those used to represent database servers 112 and 122. However, these mechanisms may include software components that are integrated within either database server 112 and 122 and that participate to transmit data along an

interconnect (e.g. network, bus) between the database servers. In fact, these transport mechanisms may use queues and tables within databases 114 and 124, or files under the control of database servers 112 and 122, and may use procedures from database transport procedures 150.

**[0027]** DB file transport mechanism 130 is a transport mechanism dedicated to transferring files between database servers 112 and 122. The files may be transported from one or more directories in a local file system of database server 112 to one or more directories of a local file system of database server 122. A local file system of a database server is a local file system of an operating system that controls the database server. Thus, a local file system of database server 112 is the local file system of the operating system of computer element 110. According to an embodiment of the present invention, the files are transported using a protocol similar to FTP.

**[0028]** DB file transport mechanism 130 is distinguishable from conventional file transport mechanisms as follows. It is a database server component that (1) only transports data in the form of files, and (2) transports a file between database servers, which either retrieve the file from a local file system and, upon receipt of the file, stores it in a local file system, and (3) transports files in this way in response to a request to transport them, where the request designates the specific files to transport and the location (e.g. directory) of the source and the destinations. The files transported maybe binary files or text files. DB file transport mechanism 130 also performs character-set conversion, converting the character-set format of the file transported.

**[0029]** While clearly transport mechanisms have transported files between file system directories, such mechanisms did not comprise software components that are integrated components of a database server. Furthermore, database servers 112 and 122, unlike

conventional database servers, are configured to recognize commands that specify files to be transported between directories. Such commands can be entered by a user via a command line interface that also accepts queries that conform to a database language, such as SQL. The commands that specify files to transport specify a file name, including a directory path of the file's location, and a file name and directory path to which to transport the file.

#### Rule-based Messaging System

[0030] Rule-based messaging system 140 transmits messages between database server 112 and database server 122. The message contains information about events, events such as the creation or modification of data. Messages are used to propagate events that occur at one database server to another database server. The other database server may then transmit the message to yet another database server.

[0031] A common use of a messaging system is to replicate data. DML changes made to a database object at a database server are propagated to another database server that maintains replicas of at least a portion of the database object. Rule-based messaging system 140 may be used to transport a wide range of types of data, including files.

[0032] The message flow needed from one messaging system to another messaging system may differ. Various types of messaging systems provide the ability for users to configure the message flow between nodes in a network. One type of messaging system is a rule-based messaging system, such as rule-based messaging system 140, which allows a user to specify rules that govern the flow of messages.

[0033] A rule specifies a condition and an action to perform if the condition is met. In general, rules comply with a rules language, which is like a computer language. Messaging systems that use rules expose information about events through variables or attributes that can be referenced by the rules. The condition in a rule may be expressed using boolean

expressions that reference the variables and attributes. The rules may be used to select which events for which messages are sent to other nodes, and what to do with a message received from another node.

### Database Transport Procedures

**[0034]** FIG. 2 depicts database transport procedures 150 in greater detail according to an embodiment of the present invention. Database transport procedures 150 include file transport procedures 210 and transportable tablespace procedures 220. File transport procedures 210 include routines related to transferring files between database servers while tablespace transport procedures 220 include procedures related to transporting tablespaces. These procedures may be invoked by a database server or user through a user interface, such as a command line interface. An invocation of a database server may specify one or more parameters and may return one or more values (e.g. function call value or parameter return values). The parameters are used to specify, for example, what tablespace to detach or which file to transport.

**[0035]** According to an embodiment of the present invention, database transport procedures 150 are written in a combination of C and PL/SQL™. PL/SQL is a procedural database language available from Oracle™ Corporation. However, the present invention is not limited to database transport procedures that are written in a particular computer language.

**[0036]** File transport procedures 210 include procedures get 212, put 214, and copy 216, as follows.

**[0037] Get 212** This procedure causes a "destination" database server to request from a "source" database server a file local to the source database server. The source database server

transmits the requested file to the destination database server. The request and the file may be transported using DB file transport mechanism 130.

**[0038] Put 214** This procedure causes a source database server to contact a destination database server to create, in the local file system of the destination database server, a copy of a file from the local file system of the source database server. The source database server transmits the requested file to the destination database server. The file may be transported using DB file transport mechanism 130. This procedure may be used by rule-base messaging system 140 to propagate a file via DB file transport mechanism 130.

**[0039] Copy 216** This procedure causes a database server to make a local copy of the file to the local file system.

**[0040] Database Transport Procedures** 150 include detach 222, clone 224, attach 226, pull 228, and push 229, as follows.

**[0041] Detach 222** This procedure causes a database server to make a set of tablespaces read-only, to detach the tablespaces from the database, and return the names of the files in the tablespace ("tablespace files"). A database server makes a set of tablespaces read-only by only honoring and processing requests to read data from the tablespaces and preventing DML changes to data in the tablespace. Tablespace metadata for the set of tablespaces is exported to a separate file, referred to as a "metadata dump file."

**[0042]** The tablespace files and the metadata dump file together form a "transportable tablespace package". A transportable tablespace package can be transported using file transport procedures 210. A transportable tablespace package detached in this way can also be "re-attached" to a database from which it was detached.

**[0043] Clone 224** This procedure copies tablespaces without detaching them. Specifically, the procedure causes a database server to make a set of tablespaces read-only, to

copy their tablespace files to another set of tablespace files that may be specified by a parameter, and then making the set of tablespaces read-write (if read-write before invoking the procedure). The procedure also exports metadata for the set of tablespaces to a metadata dump file. The procedure returns the names of the new tablespace files and the metadata dump file.

**[0044] Attach 226** This procedure causes a database server to attach a set of tablespaces in a transportable set package.

**[0045] Pull 228** This procedure causes a database server to copy a set of tablespaces from a remote database (one that is not local to the database server) of a remote database server and attach the tablespace into a local database of the database server. Thus, the procedure completely provisions a tablespace. FIG. 3 illustrates this procedure.

**[0046]** Referring to FIG. 3, at step 310 the tablespaces at the remote server are made read-only. At step 320, the database server gets metadata describing the tables using metadata import/export utilities. Such utilities allow clients and database servers to connect to another database server and get metadata describing database objects, including metadata describing tables and tablespaces and database objects within tablespaces. At step 330, the database server uses get 212 to get a copy of the tablespaces and store them in the local file system. At step 340, the tablespaces are imported into the local database by attaching them, using the tablespace metadata obtained at step 320. At step 350, the tablespaces at the remote server are made read-write.

**[0047] Push 229** This procedure causes a source database server to copy a set of tablespaces and import them into a local database of a remote database server. This procedure may be implemented by sending a message via rules-based messaging system 140 to the remote database server, the message specifying the tablespaces to "pull". The remote

database server then invokes pull 228 to provision the tablespaces in the local database. Another way to implement this procedure is for a source database server to use put 214 to transport a copy of the tablespace to a remote database, to invoke metadata import/export utilities to transmit the tablespace metadata describing the tablespaces to the remote database server, and instruct the remote database server to attach the copies using the tablespace metadata.

## AUTOMATIC PROVISIONING OF TABLESPACE SYNCHRONIZATION MECHANISMS

**[0048]** Once a copy of a tablespace is provisioned into another database, rule-based messaging system 140 may be used to keep the tablespace and its copy synchronized. To do so, various components of a rule-based messaging system 140 must be provisioned.

**[0049]** FIG. 4 is a block diagram illustrating components provisioned to synchronize a tablespace and its provisioned copy. FIG. 5 is a flowchart depicting a process for automatically provisioning a tablespace copy and rule-based messaging system components that are needed to synchronize the tablespace copy with the tablespace. These components and processes are illustrated using distributed database system 101.

**[0050]** FIG. 4 shows a rule-based messaging system 140 in greater detail, according to an embodiment of the present invention. Referring to FIG. 4, it shows that database server 112 includes a capture process 413, which captures events (e.g. DML changes and DDL changes) recorded by database server 112 in change log 417. Change log 417 contains redo and/or undo records. Capture process 413 queues messages reflecting those events into message queue 418. Propagate process 414 propagates messages from message queue 418 to message queue 422. Message queue 422 is a staging area for messages to be applied by database server 122 to database 124.

[0051] Rules engines 451 and 453 execute rules 452 and 454, respectively, in response to requests from clients to evaluate rule sets in rules 452 or 454, and then returns the evaluation results of evaluating the rule sets to the clients. Clients of rules engine 451 include capture process 413 and propagate process 414. Clients of rules engine 453 include apply process 423. Capture process 413, propagate process 414, and apply process 423 use the results of rules evaluation provided by rules engine 451 and 453 to determine how to process events and messages. This is accomplished by transmitting a request to execute rule sets to a rules engine. In response to such a request, rules engines 451 and 453 execute one or more of the rule sets. Rules-based messaging system are described in greater detail in *Utilizing Rules in Distributed Information Sharing and Utilizing Rules in a Distributed Information Sharing System*.

[0052] The process depicted in FIG. 5 is illustrated by using it to provision a copy of tablespace 419 (FIG. 4) into database 124 as tablespace 419'. Database 114 is referred to as the source database because it contains the tablespace copied and database 124 as the destination database because it is the database to which the copy is attached. Database servers 112 and 122 are referred to as the source database server and destination database server, respectively, because they are the local database servers of source database 114 and destination database 124, respectively.

[0053] Referring to FIG. 5, at step 510, source database server 112 is configured to log DML changes to tablespace 419 (i.e. changes to database objects stored in tablespace 419), to run capture process 413 to capture changes to tablespace 419, and to run propagate process 414 to propagate changes to tablespace 419 to destination database server 122.

[0054] At step 520, rules 452 are configured to cause capture process 413 to capture changes to tablespace 419 and to propagate the changes to destination database server 122 and message queue 422.

[0055] At step 530, a clone of tablespace 419, tablespace 419', is provisioned at destination database server 122 by invoking pull procedure 228.

[0056] At step 540, destination database server 122 is configured to run an apply process to apply changes from message queue 422.

[0057] At step 550, rules 454 are configured to cause apply process 423 to apply changes to tablespace 419 to tablespace 419' (i.e. to apply changes to database objects within tablespace 419 to database objects within 419').

[0058] So that the process depicted in FIG. 4 may be used to automatically and dynamically provision a tablespace, the process can be implemented as a procedure that is invoked to provision a tablespace and rule-based messaging components to a particular database or database server. Alternately, a script can be generated and later executed to provision the tablespace and/or the synchronization mechanism.

[0059] The process depicted in FIG. 4 provisions a synchronization mechanism that is unilateral, that is, changes to a tablespace are only propagated from one tablespace to another and not vice versa. However, an embodiment of the present invention is not so limited. Bilateral synchronization mechanisms may also be automatically provisioned. For example, the rules and message flow processes may be configured to propagate changes from tablespace 419' to tablespace 419 by configuring capture, propagate, and apply processes, and rules in a manner similar to that described to propagate changes from tablespace 419 to tablespace 419'.

## ADDITIONAL FEATURES OF EMBODIMENTS OF THE INVENTION

**[0060]** While the approaches to provisioning tablespaces have been illustrated by provisioning online tablespaces, the present invention is not so limited. The approaches described herein may be used to automatically and dynamically provision copies of offline tablespaces.

**[0061]** For example, an offline tablespace 180 may contain information about quarterly financial results. The data in the tablespace is only processed and reported quarterly for a brief period of time each quarter. Rather than keeping the tablespace online where it occupies resources (e.g. storage capacity), the information can be kept offline and provisioned and "unprovisioned" quarterly, freeing up resources and allowing the resources to be used for other services.

**[0062]** Although the approaches described herein provision data in the form of tablespaces, the present invention is not limited to provisioning data that is identified by tablespace. The present invention may be used to automatically provision a list of tables, schemas and databases, and a synchronization mechanism needed to keep them in sync, both bilaterally and unilaterally. Provisioning lists of tables, schemas, or databases gives customers more flexibility and a logical way to automatically provision data and keep it synchronized.

**[0063]** Synchronization mechanisms can be provisioned within distributed database systems that fall into numerous kinds of topologies. A network of database servers interlinked by messaging systems may be represented by directed graphs of nodes. Edges join the nodes, each edge representing a flow of messages from a "source" node to another adjacent "destination" node. For a given node, multiple edges can emanate from the node or terminate at the node. In addition, there can be cycles, representing messages flowing from a

source node back to the source node along a path that may include one or more other nodes. Data and/or synchronization mechanisms can be automatically provisioned in topologies that can be represented by various types of directed graphs, such as an acyclic graph or a completely connected graph, where changes between the adjacent nodes are bilaterally synchronized.

**[0064]** Nor are the approaches for automatically and dynamically provisioning data limited to provisioning database data. The approaches may be used to provision software and computer code, such as PL/SQL packages. Database servers provide code import/export utilities for exporting code to "code dump files", and for importing code from code dump files. To automatically provision code, for example, a database server could invoke a code import/export utility to export the code from a remote database server and store the code in a dump file in the local file system, and then invoke the code import/export utility to import the code into the local database or code repository. The automatic provisioning of code could be used for remote job scheduling on a remote database server. A database server could delegate a job or task to a remote database server and provide the needed code and/or data to perform the job or task by automatically provisioning the code and data.

#### USE OF OTHER FILE-BASED DATABASE PROVISIONING APPROACHES

**[0065]** Embodiments of the present invention have been illustrated using transportable tablespaces as an approach for instantiating a database, that is making a copy of a database and provisioning it. However, the present invention is not so limited. Embodiments of the present invention may use other techniques for this purpose.

**[0066]** For example, a database may be instantiated using the command generation approach. Under the command generation approach, a database server creates a file that contains a description of database objects (i.e. metadata) and database language commands

that insert data into the database objects. Such a file is referred to herein as an import/export file. The import/export file may be created using, for example, export utilities designed to export database data for selected database objects in the database by creating metadata that describes the database objects and SQL insert commands that insert rows in the database object. To provision the database data, a database server executes import utilities designed to import the data by reading the description of the database objects, ensuring that the database objects are defined in the target database (defining them if necessary), and then executing the SQL insert commands. The execution of the SQL insert commands inserts the data row-by-row into the target database, which requires more work than attaching a tablespace when inserting greater than a threshold amount of data.

**[0067]** This approach may be advantageous for provisioning a target database under various circumstances. One is when the target database already defines the database objects for the data to provision and the overhead of creating a definition of the database objects is therefore not incurred, and the amount of data to provision is small and the overhead of row-by-row insertion is therefore not significant.

**[0068]** Another circumstance is when, under the tablespace approach, a set of database objects to instantiate cannot be instantiated without having to instantiate other database objects. A tablespace is used to store data for a particular set of database objects. In some implementations of tablespaces, when using the tablespace to instantiate database objects, all the database objects in the set must be instantiated. However, under the command generation approach, it is possible to create import/export files for only a selected subset of the database objects in the set or even from other tablespaces, enabling the ability to instantiate only the selected database objects no matter which tablespace is used to store their data. Thus, the

command generation approach can provide more flexibility in forming combinations of database objects to instantiate.

**[0069]** To instantiate a copy of the database dynamically, a database server runs export utilities to create an import/export file, which is then transported between database servers using a database server file transport mechanism or messaging system, such as database server file transport mechanism 130 or rule based messaging system 140. Alternatively, import/export files may be stored as part of a library, to be transported and provisioned sometime later when needed.

**[0070]** Another approach to instantiating a database is the recovery manager approach. The recovery manager approach uses the capabilities of a recovery manager to instantiate a database. A recovery manager is used to create backup files of all database files (e.g. tablespace files) and restore database files from the backup files. Restoring the database files instantiates a copy of a database at the time of creation of the backup files. If changes to a database are archived in an archive log, the archive log may be used to restore the changes. In fact, a database can be restored to any particular point in time covered by an archive log.

**[0071]** A source database can be instantiated by using a recovery manager to create the backup files and using the backups and a recovery manager to create a target database at another location. If the backups are stored for a period time before creating the target database, and during the period time the source database changes, an archive log of the source database may be used to make the target database current or consistent with the source database at some particular point in time. In fact, it may be more efficient to instantiate multiple instances at different times from one set of backup files by using the archive log of the source database to update the multiple instances to make them consistent with a given point of time.

## HARDWARE OVERVIEW

[0072] Figure 6 is a block diagram that illustrates a computer system 600 upon which an embodiment of the invention may be implemented. Computer system 600 includes a bus 602 or other communication mechanism for communicating information, and a processor 604 coupled with bus 602 for processing information. Computer system 600 also includes a main memory 606, such as a random access memory (RAM) or other dynamic storage device, coupled to bus 602 for storing information and instructions to be executed by processor 604. Main memory 606 also may be used for storing temporary variables or other intermediate information during execution of instructions to be executed by processor 604. Computer system 600 further includes a read only memory (ROM) 608 or other static storage device coupled to bus 602 for storing static information and instructions for processor 604. A storage device 610, such as a magnetic disk or optical disk, is provided and coupled to bus 602 for storing information and instructions.

[0073] Computer system 600 may be coupled via bus 602 to a display 612, such as a cathode ray tube (CRT), for displaying information to a computer user. An input device 614, including alphanumeric and other keys, is coupled to bus 602 for communicating information and command selections to processor 604. Another type of user input device is cursor control 616, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processor 604 and for controlling cursor movement on display 612. This input device typically has two degrees of freedom in two axes, a first axis (e.g., x) and a second axis (e.g., y), that allows the device to specify positions in a plane.

[0074] The invention is related to the use of computer system 600 for implementing the techniques described herein. According to one embodiment of the invention, those

techniques are performed by computer system 600 in response to processor 604 executing one or more sequences of one or more instructions contained in main memory 606. Such instructions may be read into main memory 606 from another computer-readable medium, such as storage device 610. Execution of the sequences of instructions contained in main memory 606 causes processor 604 to perform the process steps described herein. In alternative embodiments, hard-wired circuitry may be used in place of or in combination with software instructions to implement the invention. Thus, embodiments of the invention are not limited to any specific combination of hardware circuitry and software.

**[0075]** The term “computer-readable medium” as used herein refers to any medium that participates in providing instructions to processor 604 for execution. Such a medium may take many forms, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media includes, for example, optical or magnetic disks, such as storage device 610. Volatile media includes dynamic memory, such as main memory 606. Transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus 602. Transmission media can also take the form of acoustic or light waves, such as those generated during radio-wave and infra-red data communications.

**[0076]** Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, or any other magnetic medium, a CD-ROM, any other optical medium, punchcards, papertape, any other physical medium with patterns of holes, a RAM, a PROM, and EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave as described hereinafter, or any other medium from which a computer can read.

**[0077]** Various forms of computer readable media may be involved in carrying one or more sequences of one or more instructions to processor 604 for execution. For example, the instructions may initially be carried on a magnetic disk of a remote computer. The remote

computer can load the instructions into its dynamic memory and send the instructions over a telephone line using a modem. A modem local to computer system 600 can receive the data on the telephone line and use an infra-red transmitter to convert the data to an infra-red signal. An infra-red detector can receive the data carried in the infra-red signal and appropriate circuitry can place the data on bus 602. Bus 602 carries the data to main memory 606, from which processor 604 retrieves and executes the instructions. The instructions received by main memory 606 may optionally be stored on storage device 610 either before or after execution by processor 604.

**[0078]** Computer system 600 also includes a communication interface 618 coupled to bus 602. Communication interface 618 provides a two-way data communication coupling to a network link 620 that is connected to a local network 622. For example, communication interface 618 may be an integrated services digital network (ISDN) card or a modem to provide a data communication connection to a corresponding type of telephone line. As another example, communication interface 618 may be a local area network (LAN) card to provide a data communication connection to a compatible LAN. Wireless links may also be implemented. In any such implementation, communication interface 618 sends and receives electrical, electromagnetic or optical signals that carry digital data streams representing various types of information.

**[0079]** Network link 620 typically provides data communication through one or more networks to other data devices. For example, network link 620 may provide a connection through local network 622 to a host computer 624 or to data equipment operated by an Internet Service Provider (ISP) 626. ISP 626 in turn provides data communication services through the world wide packet data communication network now commonly referred to as the "Internet" 628. Local network 622 and Internet 628 both use electrical, electromagnetic

or optical signals that carry digital data streams. The signals through the various networks and the signals on network link 620 and through communication interface 618, which carry the digital data to and from computer system 600, are exemplary forms of carrier waves transporting the information.

**[0080]** Computer system 600 can send messages and receive data, including program code, through the network(s), network link 620 and communication interface 618. In the Internet example, a server 630 might transmit a requested code for an application program through Internet 628, ISP 626, local network 622 and communication interface 618.

**[0081]** The received code may be executed by processor 604 as it is received, and/or stored in storage device 610, or other non-volatile storage for later execution. In this manner, computer system 600 may obtain application code in the form of a carrier wave.

**[0082]** In the foregoing specification, embodiments of the invention have been described with reference to numerous specific details that may vary from implementation to implementation. Thus, the sole and exclusive indicator of what is the invention, and is intended by the applicants to be the invention, is the set of claims that issue from this application, in the specific form in which such claims issue, including any subsequent correction. Any definitions expressly set forth herein for terms contained in such claims shall govern the meaning of such terms as used in the claims. Hence, no limitation, element, property, feature, advantage or attribute that is not expressly recited in a claim should limit the scope of such claim in any way. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.